**Review the Program Specification**

Refer to the Journal program specification. As a team, review the program requirements and how it is supposed to work.

1. What does the program do?
    - Provide a text menu. Number selects menu option
        - Write - #1
            - Selects from random prompts and displays to user
            - Takes input from user.
        - Display - #2
            - Display what's been written.
            - Shows
                - Date, prompt given and response
                - Shows all entries in the journal
        - Load - #3
            - Prompts user for a filename
            - Loads the file
                - Allows "display" option to show everything in that file.
                - Allows further "write" entries to append to the file.
        - Save - #4
            - Prompts user for a filename
            - Save to a file.
        - Quit - #5
            - Quit the program
2. What user inputs does it have?
    - Menu option (typed number of which menu option) (integer)
    - Answers to the journal prompts (string) - #1
    - Filename to load (string) - #3
    - Filename to save (string) - #4
3. What output does it produce?
    - Menu
    - Prompts for journal questions - #1
    - Display of all journal entries in memory (loaded from textfile + entries not yet saved) - #2
    - Prompt for filename to load. - #3
    - Prompt for filename to save. - #4
4. How does the program end?
    - User typing #5 from menu.

**Determine the classes**

The first step in designing a program like this is to think about the classes you will need. When thinking about classes, it is often helpful to consider the strong nouns in the program description.

1. What are good candidates for classes in this program?
2. What are the primary responsibilities of each class?
    - **Journal** (collection of prompts + Entries)
        - store all timestamps, prompts & entries from textfile (if loaded)
        - Return all timestamps, prompts & entries from textfile + entries given by user not yet saved to file. (used to display back to user)
    - **Prompt** (random prompts)
        - Random selection of prompt
        - Return selected prompt (for Display)
        - Store pre-defined prompts
    - **Entry** (user entries/responses to random prompts)
        - Collect journal entries.
        - Get Date

**Define class behaviors**

Now that you have decided on the classes, you will need and their responsibilities, the next step is to define the **behaviors (functions)** of these classes. These will become methods for the class.

Go through each of your classes and ask:

1. What are the behaviors this class will have in order to fulfill its responsibilities? (In other words, what things should this class do?)
    - **Journal:**
        - Add Entry
        - Get Entries
        - Save to File
        - Load from File
        - Display back Entries
    - **Prompt:**
        - Return random selection of stored prompts
    - **Entry:**
        - ~~Collect User Name (optional)~~
        - Collect Date

Recommended Video:

- CSV Reading and Writing Demo (20 M

**Class Diagrams:**

**Class: Journal**

Attributes:
- _entries: list
- _filename: string

Behaviors:
- AddEntry(): void
- GetEntries(): void
- SaveToFile(): void
- LoadFromFile(): void
- DisplayEntries(): void

**Class: Prompt**

Attributes:
- _prompts: list

Behaviors:
- RandomPrompt(): string

**Class Entry**

Attributes:
- _username: string
- _date: string
- _prompt: string
- _userResponse: string

Behaviors:
- GetUserName(): string
- GetDate(): string
- GetPrompt(): string
- GetResponse(): string
- BuildEntry(): string
- DisplayEntries(): void

- Collect Prompt
- Collect user Response
- Build Entry (Date, Prompt, Response)
- Display back Entries

**Define class attributes**

Now that you have defined the classes, their responsibilities, and their behaviors, the next step is to determine what **attributes (variables)** the class should have, or what variables it needs to store.

Go through each of your classes and ask:

1. What attributes does this class need to fulfill its behaviors? (In other words, what variables should this class store?)
2. What are the data types of these member variables?
    - **Journal:**
        - _entries:  list
        - _filename: string
    - **Prompt:**
        - _prompts: list
    - **Entry:**
        - _username: string
        - _date: string
        - _prompt: string
        - _userResponse: string